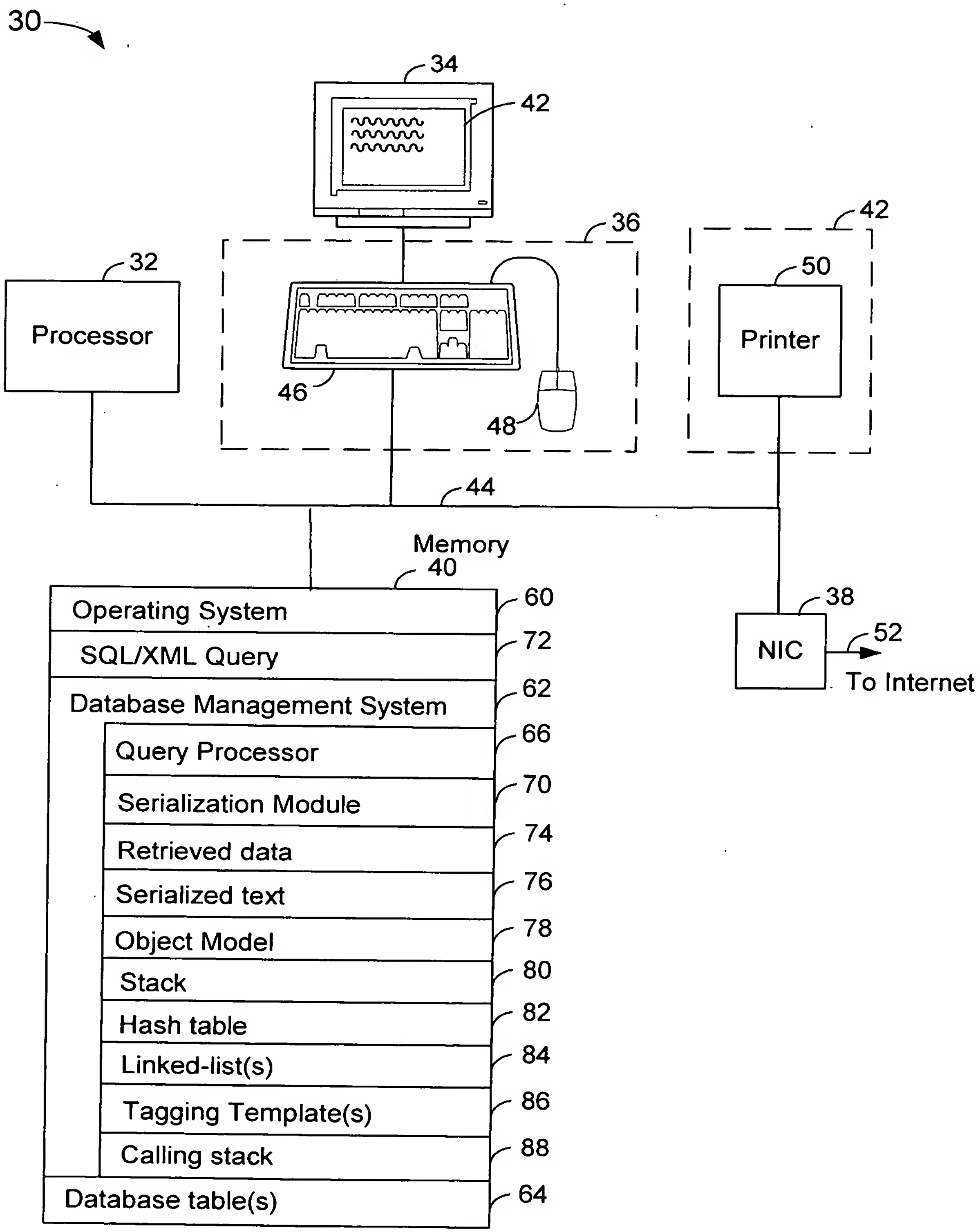
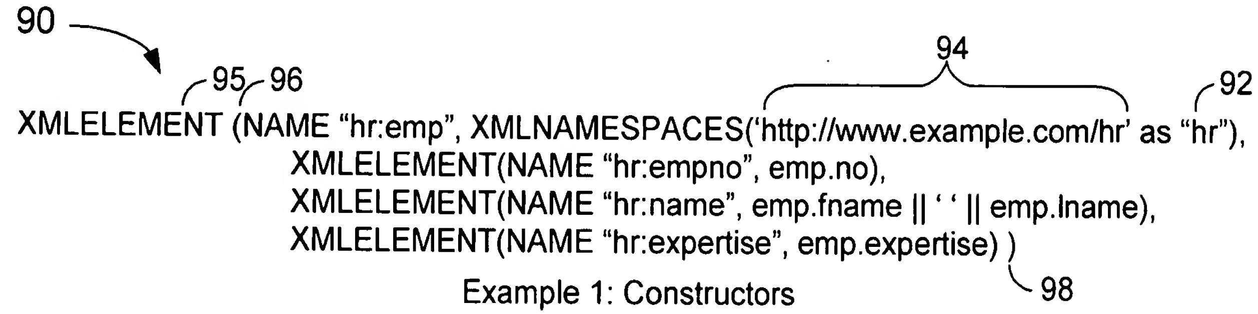

30
34
42
36
42
32
Processor
50
Printer
46
48
44
Memory
40
Operating System
60
SQL/XML Query
72
Database Management System
62
Query Processor
66
Serialization Module
70
Retrieved data
74
Serialized text
76
Object Model
78
Stack
80
Hash table
82
Linked-list(s)
84
Tagging Template(s)
86
Calling stack
88
Database table(s)
64
38
NIC
52
To Internet

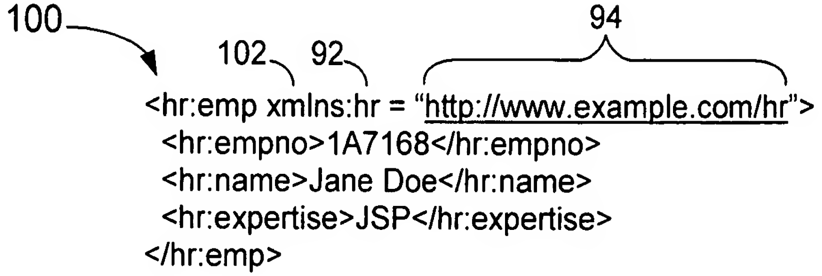
FIG. 1



```
XMLELEMENT (NAME "hr:emp", XMLNAMESPACES('http://www.example.com/hr' as "hr"),
                  XMLELEMENT(NAME "hr:empno", emp.no),
                  XMLELEMENT(NAME "hr:name", emp.fname || ' ' || emp.lname),
                  XMLELEMENT(NAME "hr:expertise", emp.expertise) )
```

Example 1: Constructors

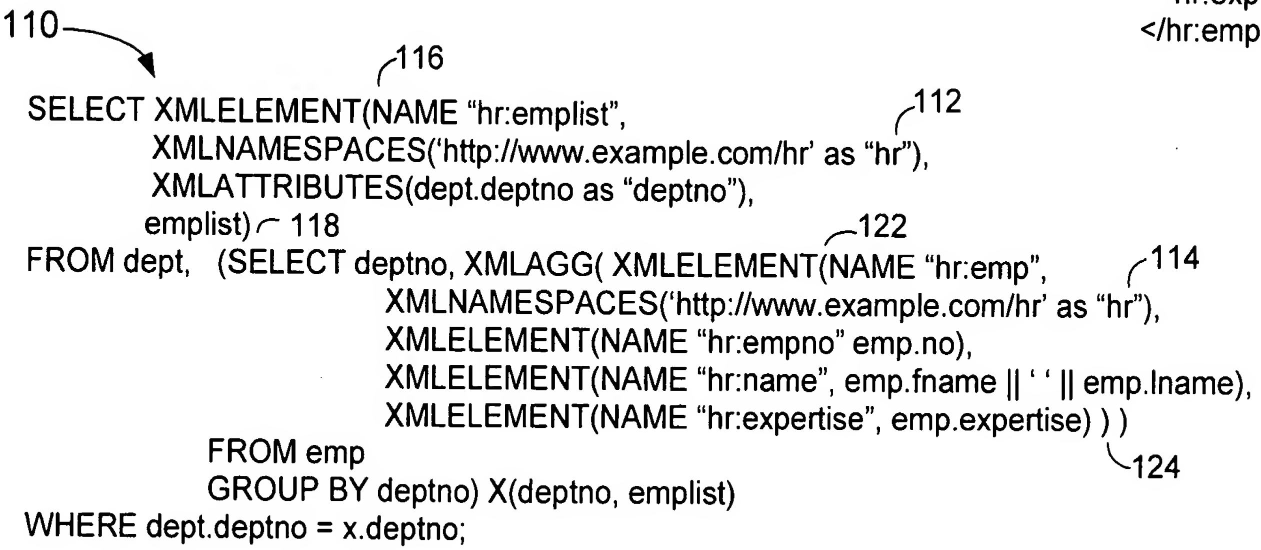
FIG. 2



```
<hr:emp xmlns:hr = "http://www.example.com/hr">
  <hr:empno>1A7168</hr:empno>
  <hr:name>Jane Doe</hr:name>
  <hr:expertise>JSP</hr:expertise>
</hr:emp>
```

Example 1: Serialized XML text

FIG. 3



```
SELECT XMLELEMENT(NAME "hr:emplist",
       XMLNAMESPACES('http://www.example.com/hr' as "hr"),
       XMLATTRIBUTES(dept.deptno as "deptno"),
       emplist)
FROM dept,   (SELECT deptno, XMLAGG( XMLELEMENT(NAME "hr:emp",
                        XMLNAMESPACES('http://www.example.com/hr' as "hr"),
                        XMLELEMENT(NAME "hr:empno" emp.no),
                        XMLELEMENT(NAME "hr:name", emp.fname || ' ' || emp.lname),
                        XMLELEMENT(NAME "hr:expertise", emp.expertise) ) )
            FROM emp
            GROUP BY deptno) X(deptno, emplist)
WHERE dept.deptno = x.deptno;
```

Example 2: Query

FIG. 4

130

132

134

136

138

```
<hr:emplist xmlns:hr = “http://www.example.com/hr” deptno=“5”>
  <hr:emp xmlns:hr = “http://www.example.com/hr”>
    <hr:empno>1BC245</hr:empno>
    <hr:name>John Doea </hr:name>
    <hr:expertise>C/C++</hr:expertise>
  </hr:emp>
  <hr:emp xmlns:hr = “http://www.example.com/hr”>
    <hr:empno>1A7168</hr:empno>
    <hr:name>Jane Doe</hr:name>
    <hr:expertise>JSP</hr:expertise>
  </hr:emp>
  <hr:emp xmlns:hr = “http://www.example.com/hr”>
    <hr:empno>1C8N12</hr:empno>
    <hr:name>James Doeb</hr:name>
    <hr:expertise>Meditation</hr:expertise>
  </hr:emp>
…
</hr:emplist>
```

Prior Art

Example 2: Exemplary serialized XML text

FIG. 5

150

```
<emp xmlns = “http://www.example.com/hr”>
  <empno>1A7168</empno>
  <name>Jane Doe</name>
  <expertise>JSP</expertise>
</emp>
```

Example 3: Exemplary serialized XML text

FIG. 7

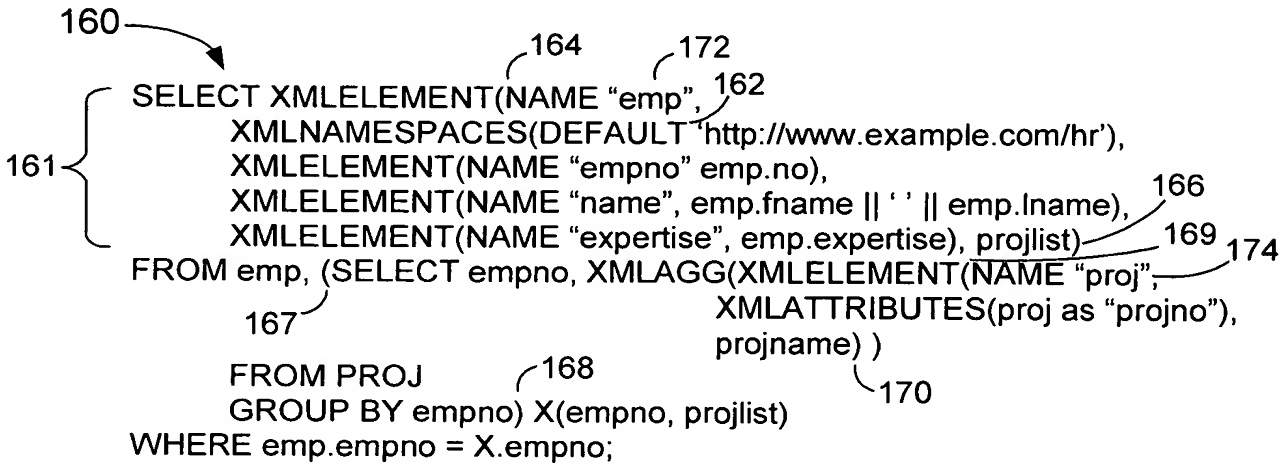
140

142

```
XMLELEMENT(NAME “emp”, XMLNAMESPACES(DEFAULT ‘http://www.example.com/hr’),
               XMLELEMENT(NAME “empno” emp.no),
               XMLELEMENT(NAME “name”, emp.fname || ‘ ’ emp.lname),
               XMLELEMENT(NAME “expertise”, emp.expertise) )
```

Example 3: Constructors

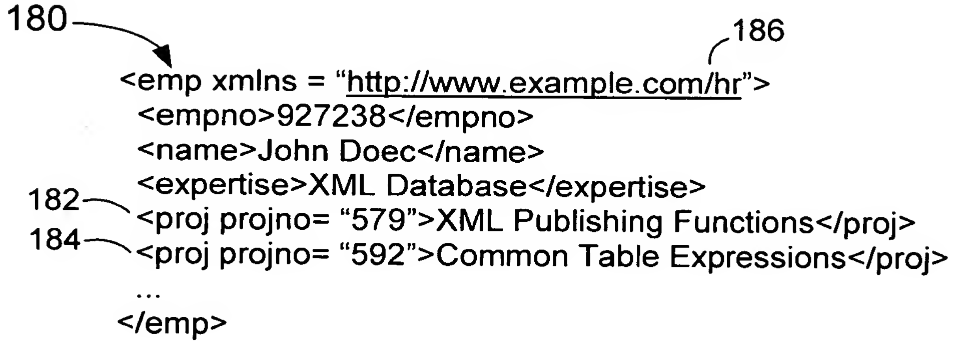
FIG. 6



```
SELECT XMLELEMENT(NAME "emp",
       XMLNAMESPACES(DEFAULT 'http://www.example.com/hr'),
       XMLELEMENT(NAME "empno" emp.no),
       XMLELEMENT(NAME "name", emp.fname || ' ' || emp.lname),
       XMLELEMENT(NAME "expertise", emp.expertise), projlist)
FROM emp, (SELECT empno, XMLAGG(XMLELEMENT(NAME "proj",
                                   XMLATTRIBUTES(proj as "projno"),
                                   projname) )
       FROM PROJ
       GROUP BY empno) X(empno, projlist)
WHERE emp.empno = X.empno;
```

Example 4: Query

FIG. 8

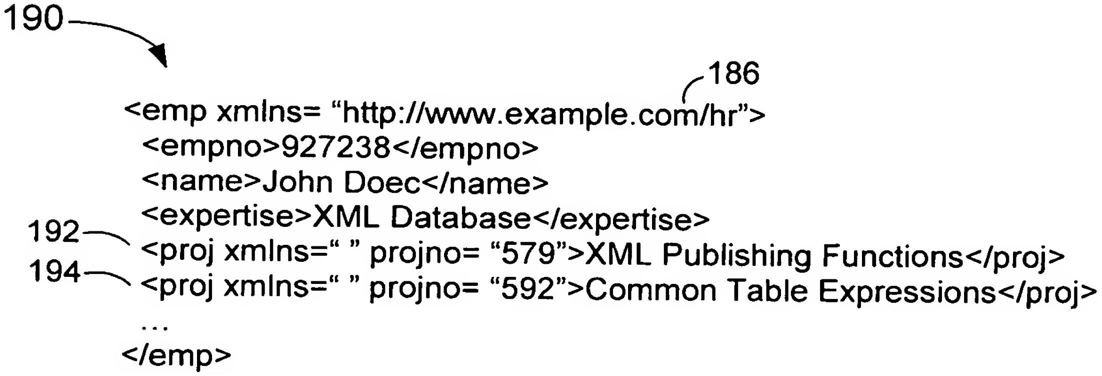


```
<emp xmlns = "http://www.example.com/hr">
  <empno>927238</empno>
  <name>John Doec</name>
  <expertise>XML Database</expertise>
  <proj projno= "579">XML Publishing Functions</proj>
  <proj projno= "592">Common Table Expressions</proj>
  ...
</emp>
```

Example 4a: Serialized XML text generated using a conventional technique

FIG. 9

Prior Art



```
<emp xmlns= "http://www.example.com/hr">
  <empno>927238</empno>
  <name>John Doec</name>
  <expertise>XML Database</expertise>
  <proj xmlns=" " projno= "579">XML Publishing Functions</proj>
  <proj xmlns=" " projno= "592">Common Table Expressions</proj>
  ...
</emp>
```

Example 4b: Serialized XML text

FIG. 10

200

```
SELECT XMLELEMENT(NAME “emp”,
        XMLNAMESPACES(DEFAULT ‘http://www.example.com/hr’),
        XMLELEMENT(NAME “empno” emp.no),
        XMLELEMENT(NAME “name”, emp.fname || ‘ ’ || emp.lname),
        XMLELEMENT(NAME “expertise”, emp.expertise),
        projname)  )
FROM emp, (SELECT empno, XMLAGG(XMLELEMENT(NAME “proj”,
                                XMLNAMESPACES(NO DEFAULT), 202
                                XMLATTRIBUTES(projno as “projno”),
                                projname) )
        FROM PROJ
        GROUP BY empno) X(empno, projlist)
WHERE emp.empno = X.empno);
```

Example 5: Modified SQL/XML query to force conventional technique to generate correct serialized XML text

FIG. 11

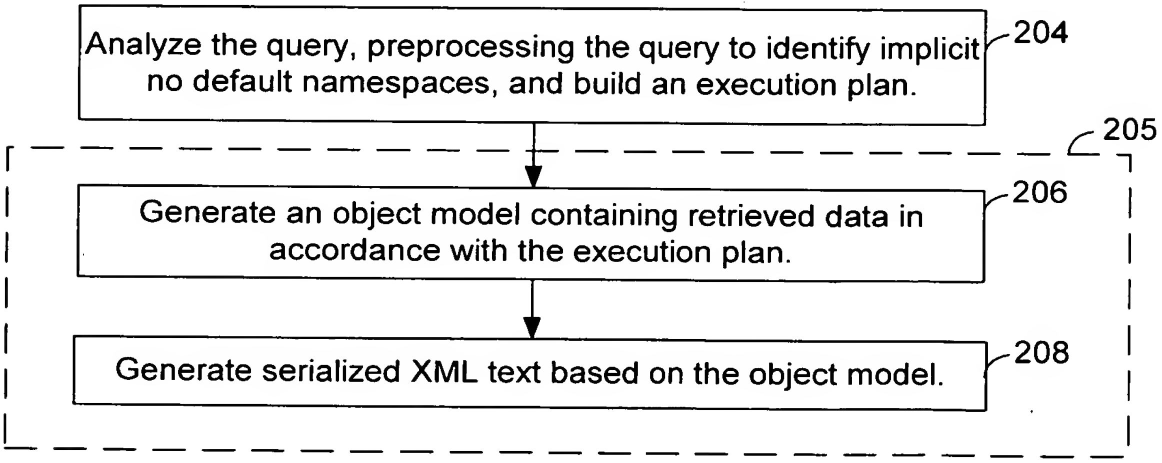
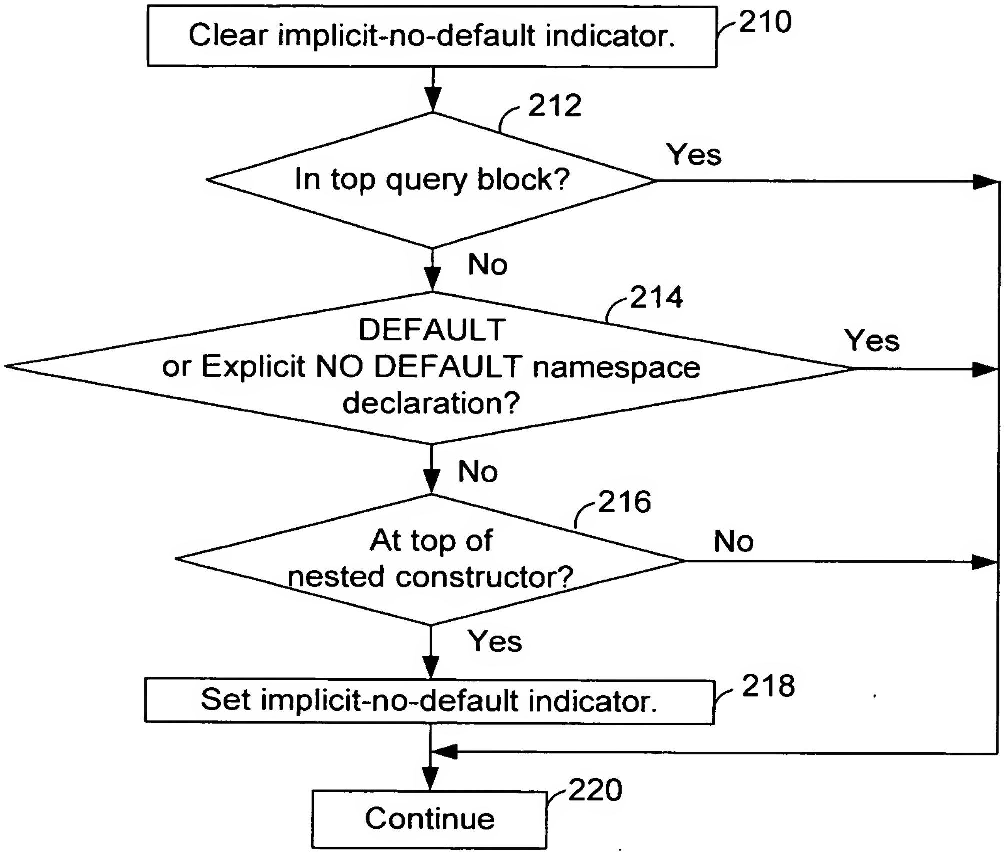

Analyze the query, preprocessing the query to identify implicit no default namespaces, and build an execution plan.
204
205
Generate an object model containing retrieved data in accordance with the execution plan.
206
Generate serialized XML text based on the object model.
208

FIG. 12



Pre-processing

FIG. 13

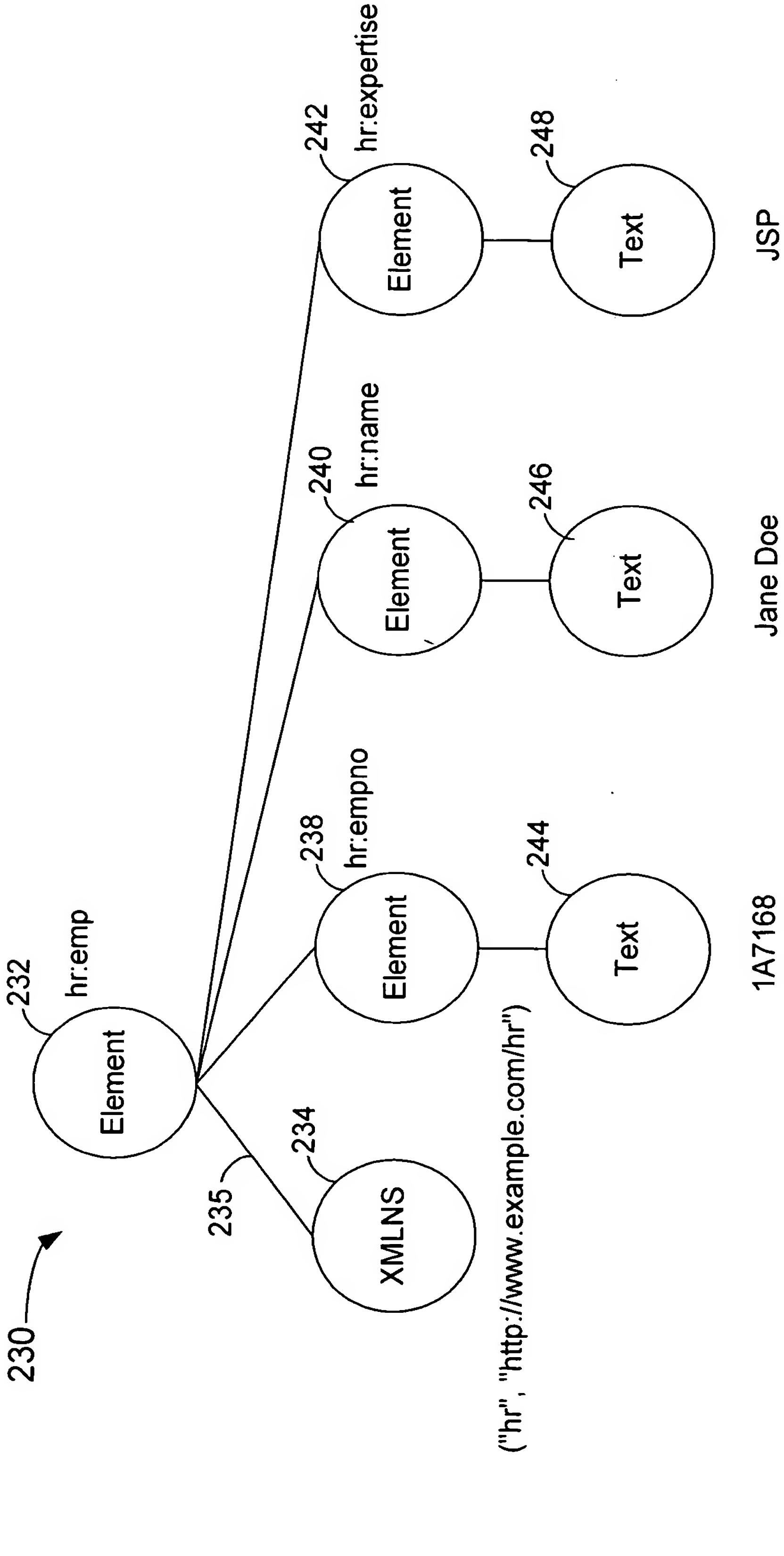

230
232
hr:emp
Element
235
234
XMLNS
("hr", "http://www.example.com/hr")
238
hr:empno
Element
244
Text
1A7168
240
hr:name
Element
246
Text
Jane Doe
242
hr:expertise
Element
248
Text
JSP

FIG. 14

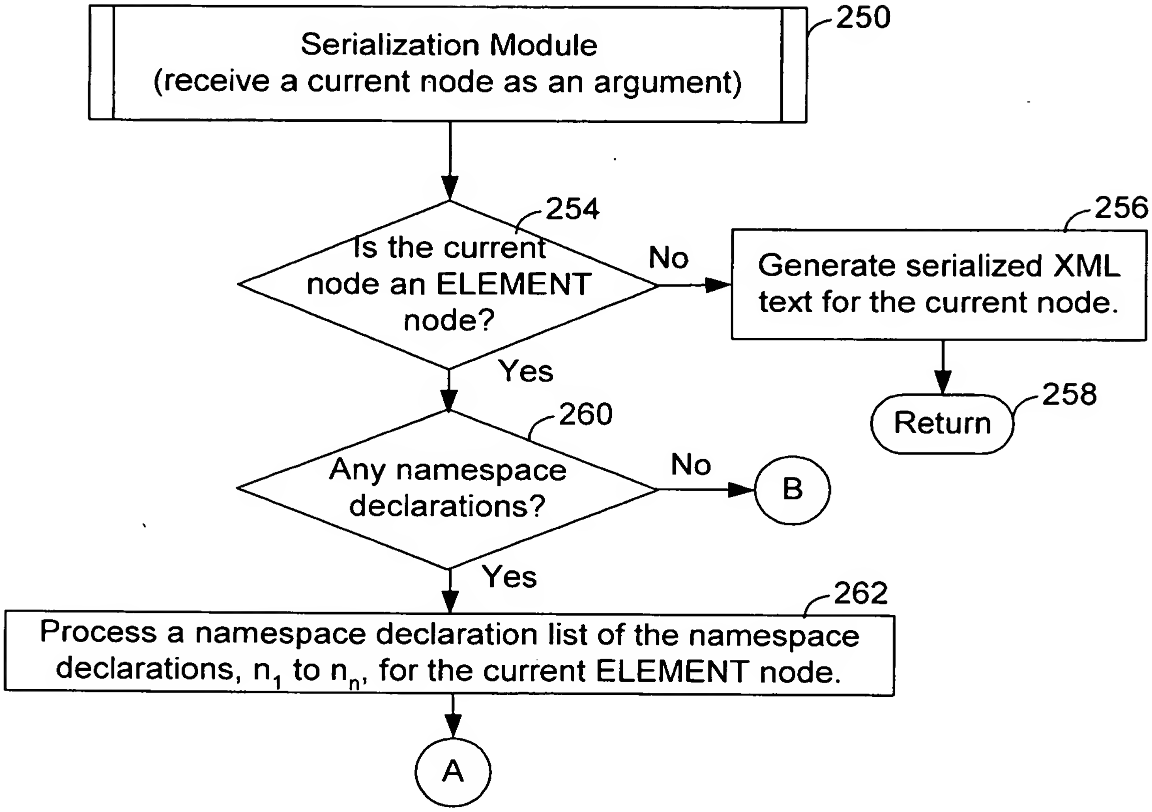

Serialization Module
(receive a current node as an argument)
250
254
Is the current node an ELEMENT node?
No
256
Generate serialized XML text for the current node.
Yes
258
Return
260
Any namespace declarations?
No
B
Yes
262
Process a namespace declaration list of the namespace declarations, n_1 to n_n, for the current ELEMENT node.
A

FIG. 15A

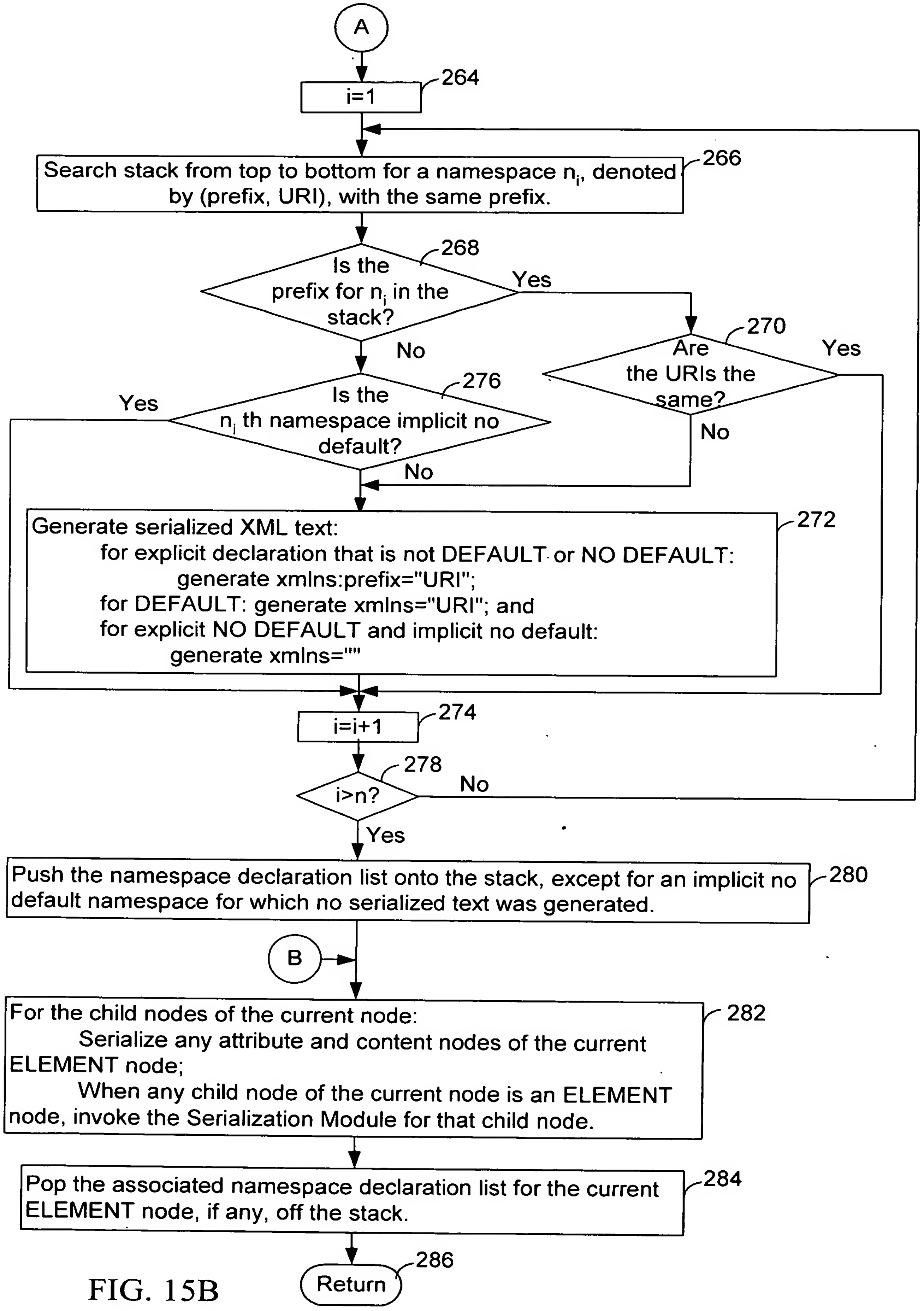

A
i=1
264
Search stack from top to bottom for a namespace n_i, denoted by (prefix, URI), with the same prefix.
266
Is the prefix for n_i in the stack?
268
Yes
No
Are the URIs the same?
270
Yes
No
Is the n_i th namespace implicit no default?
276
Yes
No
Generate serialized XML text:
for explicit declaration that is not DEFAULT or NO DEFAULT:
generate xmlns:prefix="URI";
for DEFAULT: generate xmlns="URI"; and
for explicit NO DEFAULT and implicit no default:
generate xmlns=""
272
i=i+1
274
i>n?
278
No
Yes
Push the namespace declaration list onto the stack, except for an implicit no default namespace for which no serialized text was generated.
280
B
For the child nodes of the current node:
Serialize any attribute and content nodes of the current ELEMENT node;
When any child node of the current node is an ELEMENT node, invoke the Serialization Module for that child node.
282
Pop the associated namespace declaration list for the current ELEMENT node, if any, off the stack.
284
Return
286

FIG. 15B

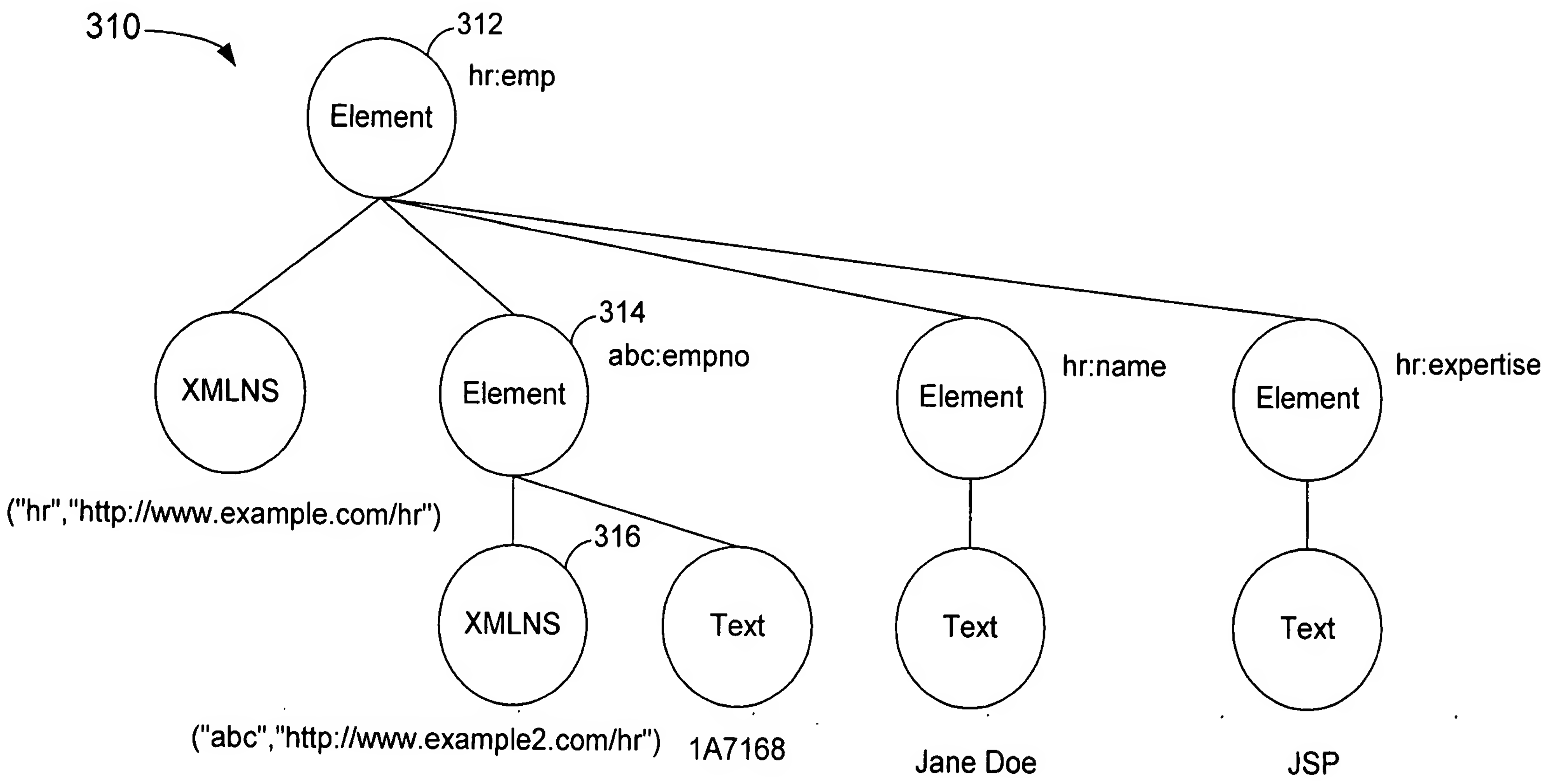
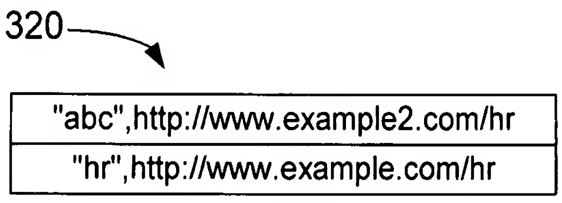


FIG. 16



"abc",http://www.example2.com/hr
"hr",http://www.example.com/hr

Stack

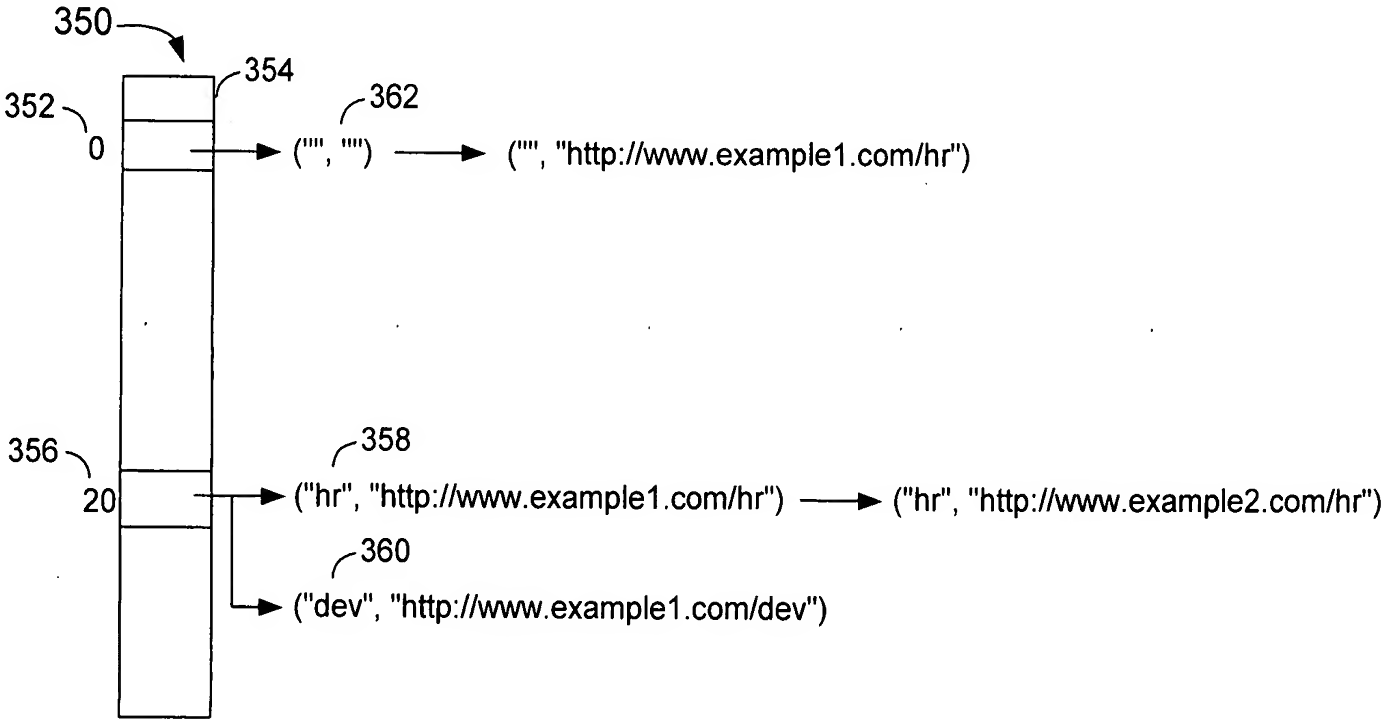
FIG. 17

330

```
XMLELEMENT(NAME “hr:emp”, XMLNAMESPACES(‘http://www.example.com/hr’ as “hr”),
        XMLATTRIBUTES(emp.no AS “hr.empno”),
        XMLELEMENT(NAME “hr:name”, emp.fname || ‘ ’ || emp.lname),
        XMLELEMENT(NAME “hr:expertise”, emp.expertise ) )
```

Example 6: Constructors

FIG. 18


350
354
352
0
362
("", "")
("", "http://www.example1.com/hr")
356
20
358
("hr", "http://www.example1.com/hr")
("hr", "http://www.example2.com/hr")
360
("dev", "http://www.example1.com/dev")

Hash table anchor array

FIG. 20

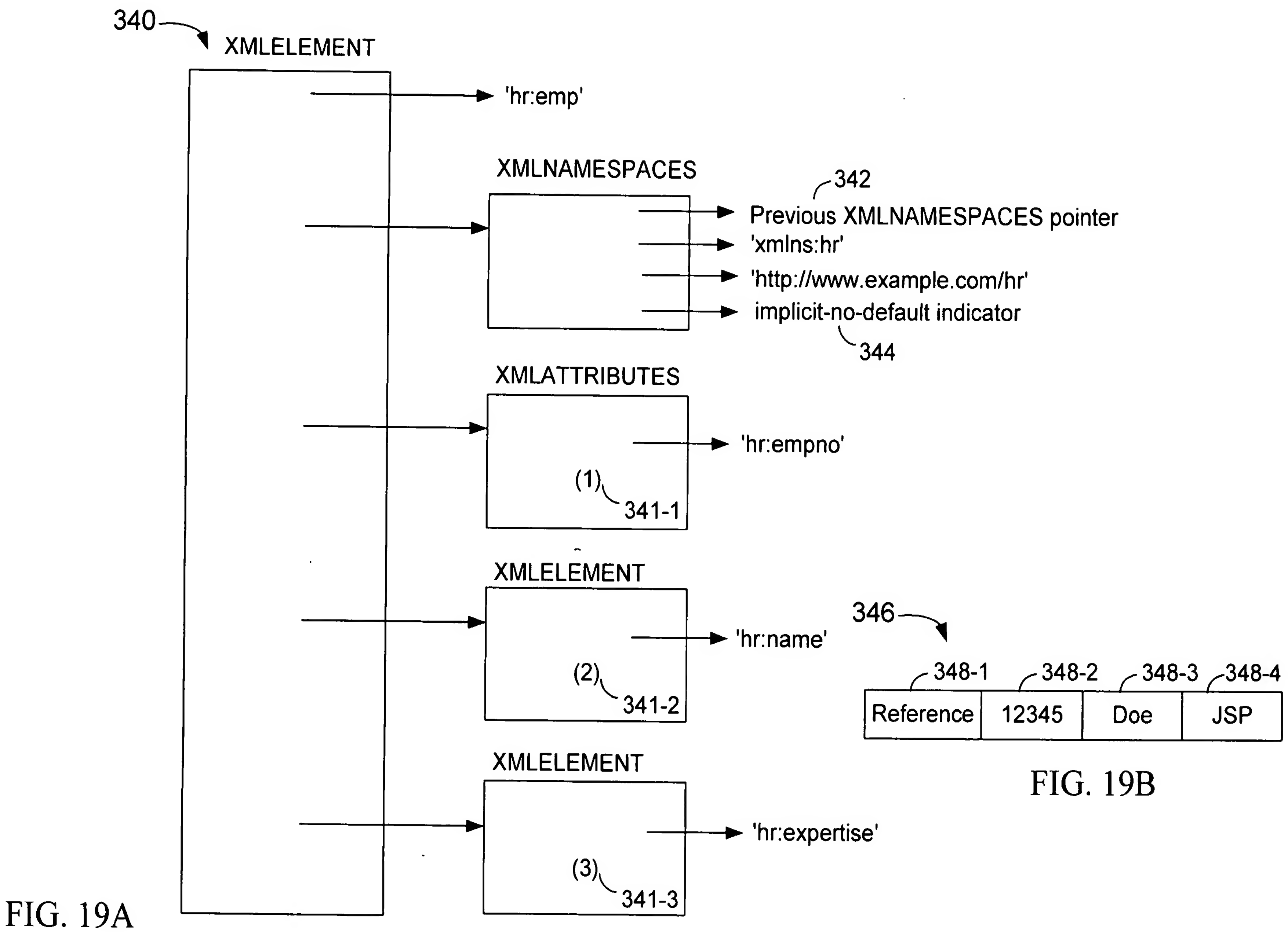


FIG. 19A

FIG. 19B

350

```
WITH MANAGE(I, MgrNo, Deptno, Deptname, EmpXMLList) as
(SELECT 1, MgrNo, Deptno, Deptname,
      XMLAGG( XMLELEMENT(NAME "x:Emp",
             XMLNAMESPACES('http://www.example.com/x' as "x"),
             EMP.EmpName) )
 FROM   DEPT, EMP
 WHERE  DEPT.Detpno = EMP.Deptno
 GROUP BY MgrNo, Deptno, Deptname
UNION ALL                                                          352
 SELECT I+1, D1.MgrNo, D1.Deptno, D1.Deptname,     --- recursion part 1
      XMLELEMENT(NAME "y:Dept",
        XMLNAMESPACES('http://www.example.com/y' as "y"),
        XMLATTRIBUTES(EMP.DeptNo as "MgrDept"),
         M.EmpXMLList )
 FROM   EMP, MANAGE M
 WHERE  M.MgrNo = EMP.empno AND
      I < 10
UNION ALL                                                        354
 SELECT I+1, M.mgrNo, D.Deptno, D.Deptname,        --- recursion part 2
      XMLELEMENT(NAME "z:Dept",
       XMLNAMESPACES('http://www.example.com/z' as "z"),
       XMLATTRIBUTES (D.Deptname as "DeptName"),
       M.EmpXMLList )
 FROM   DEPT D, MANAGE M
 WHERE  D.Deptno = M.Deptno AND I < 10
)
SELECT XMLELEMENT(NAME "x:Mgr",
      XMLNAMESPACES('http://www.example.com/x' as "x"),
      XMLATTRIBUTES(MgrNo as "MgrNo"),
      EMPXMLList)
FROM MANAGE
WHERE MgrNo = '101';
```

Exemplary SQL/XML query with recursion

FIG. 21

360

```
struct xmlnsentry
{                            364
  xmlnsentry   * prev;                362
  xmlnamespaces  * xmlnsptr;
} myxmlnsentry;
```

FIG. 22

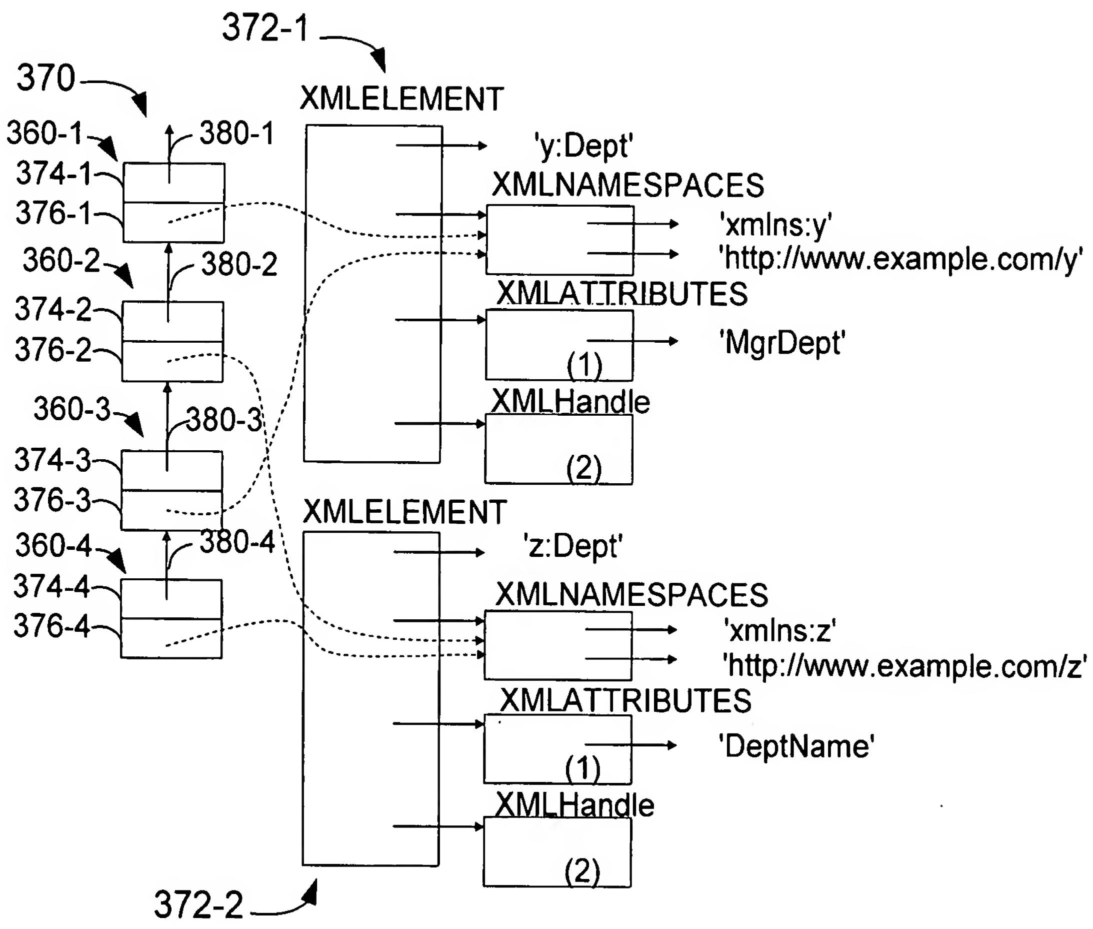

372-1
370
XMLELEMENT
360-1
380-1
374-1
376-1
'y:Dept'
XMLNAMESPACES
'xmlns:y'
'http://www.example.com/y'
360-2
380-2
374-2
376-2
XMLATTRIBUTES
(1)
'MgrDept'
360-3
380-3
XMLHandle
(2)
374-3
376-3
XMLELEMENT
360-4
380-4
'z:Dept'
374-4
376-4
XMLNAMESPACES
'xmlns:z'
'http://www.example.com/z'
XMLATTRIBUTES
(1)
'DeptName'
XMLHandle
(2)
372-2

FIG. 23